Departement
Industriële Wetenschappen &
Technologie

# Intelligent sensor for substation monitoring

UC Leuven Limburg
MOVING MINDS

Stagerapport van

**Frederik Vandeweyer**

Kandidaten voor de graad van Professionele Bachelor in de opleiding Elektronica-ICT

afstudeerrichting Elektronica

**Promotoren:**

Ladislav Šťastný

Koen Naelaerts

Academiejaar: 2014-2015
Referentie: E15_S_ELO-ICT_ELO_07

Departement
Industriële Wetenschappen & Technologie

# Intelligent sensor for substation monitoring

Stagerapport van

**Frederik Vandeweyer**

Kandidaten voor de graad van Professionele Bachelor in de opleiding Elektronica-ICT

afstudeerrichting Elektronica

**Promotoren:**

Ladislav Šťastný

Koen Naelaerts

Academiejaar: 2014-2015
Referentie: E15_S_ELO-ICT_ELO_07

# Foreword

I've lived in the Czech Republic for almost 5 months and did my Internship at the Brno University of Technology. This was an amazing end to my electronics studies and I want to thank several people who made this possible.

Ms. Greet Raymaekers for suggesting the possibility and making it all possible. And of course her Czech counterpart Ms. Jarmila Poncova who very quickly found a place for me in Brno.

Our International coordinator Mr. Wim Claes, without his help none of the required paperwork would have ever made it on time.

And of course my two promoters: Mr Koen Naelaerts, my internal supervisor, for his support in Belgium and for teaching me most things in the first place.
And Mr. Ladislav Stastny, my supervisor in Brno, without his help and support none of it would have ever been possible.

# Table of Contents

# Abstract

This thesis is part of the international ERASMUS exchange program at the Brno University of Technology in the Czech Republic.

It is the first step for a larger project to modernize electrical substations and have all equipment within it communicate using a Modbus interface. This first part focuses on making sensors for data collection.

I designed a prototype PCB with sensors for ozone, temperature and humidity which can communicate its findings over a Modbus interface. Each of these measurements acts as an early warning of failure in equipment in electrical substations.

Humidity should be kept low to keep equipment in good working condition. High humidity also increases the danger of electrical arcs which creates an unacceptably unsafe environment for anyone doing maintenance work.

Ozone is produced as an immediate effect of corona or surface predischarge phenomena. In the long term they lead to the breakdown of equipment which would result in the substation being out of service.

Lastly the high-voltage transformer shouldn't exceed its safe operational temperature. An infrared thermometer is used to measure its temperature without risking any wires being close to the device.

All these measurements can be read over a Modbus interface to alert maintenance personnel or even shutdown the transformer before there is any permanent damage.

This was a very interesting project that dealt with every aspect of electronic design. From finding good quality but affordable sensors, designing a working board and programming it.

# PART 1: Research

# 1 Introduction

There are 3 sensors involved in the design: IR temperature sensor, Ozone sensor and humidity sensor.

To ensure the correct operation of an electrical substation it is useful to keep track of the temperature in the transformer. An InfraRed Temperature sensor (FIR sensor) allows us to monitor the temperature from a safe distance.
There are 2 main considerations to take into account when selecting a FIR sensor. The temperature range and the field of view of the sensor.

An early warning of failure in equipment in an electrical substation is to detect corona or surface predischarges phenomena. In the long term they lead to the breakdown of equipment which would result in the substation being out of service. To detect this we can look for the immediate effects of these phenomena such as light emission, acoustic noise and ozone production.
Of these 3 effects, ozone is easy to measure. We can have our micro-controller look for a (relatively) fast increase in ozone concentrations as an indication, while ignoring slow changing background levels.

An other important parameter to measure is air humidity. Humidity in an electrical substation should be low to keep the equipment in good working condition. And a high humidity also increases the danger of electrical arcs.

The goal of the research phase is to find suitable and affordable sensors for use in the design of a prototype. This is done by reading datasheets, comparing specifications, contacting manufacturers and finding distributors for candidate parts.

# 2 Temperature sensor

## 2.1 Theory

### 2.1.1 Basic operation

Everything with a temperature above absolute zero emits radiation that depends mainly on its temperature, we call it heat radiation. Most of this radiation is infrared and outside of the visible spectrum.

In the same way as with wireless radio transmissions where energy is emitted by a transmitter and captured by a receiver via an antenna and transformed into useful information, the heat radiating from an object is received by a sensor and transformed into electrical signals.

A sensors capable of measuring temperature at a distance are called: radiation thermometers, radiation pyrometers or simply pyrometers.

Measuring temperature this way has several advantages:
- fast response time
- interference-free measurements
- capability to measure moving objects
- capability to measuring difficult or impossible to reach objects

In addition to emitting radiation because of its own heat, an object can also reflect radiation from other objects or let radiation pass right through it.

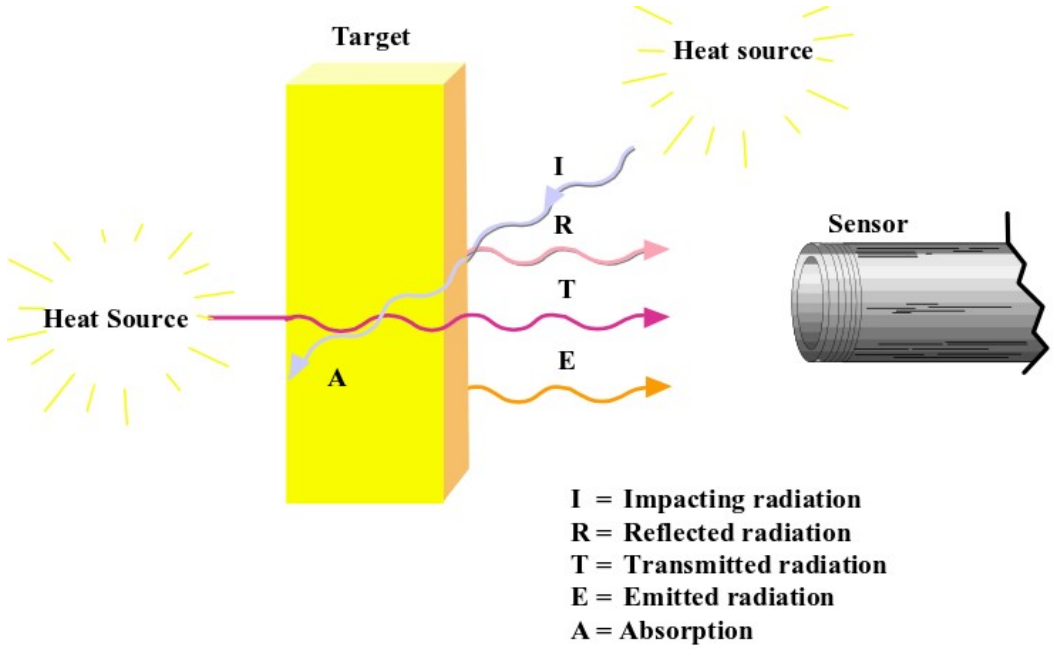


*Image 1: radiation within field of view of the sensor*

It is necessary to take all this energy into account when calculating the temperature of an object based on the radiation received by the sensor.

### 2.1.2 Field of view

A FIR sensor will measure the average temperature of everything within its field of view. Because of this the object we want to measure must take up the entire field of view of the sensor.

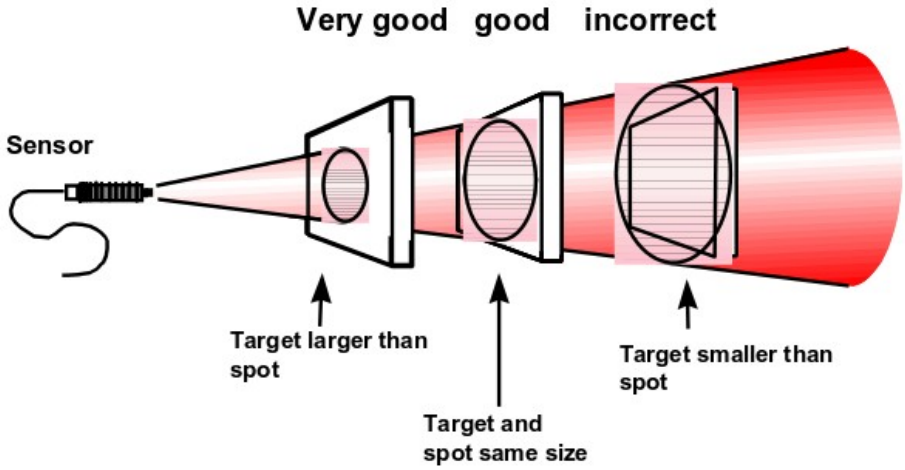


*Image 2: Target placement*

The field of view is defined as the relationship between the distance of the sensor from the target and the diameter of the spot (D:S). The greater this value the better the optical resolution of the measuring device and the smaller the target can be at a given distance.

Alternatively the field of view may be given by the angle towards the target.

### 2.1.3 Temperature range

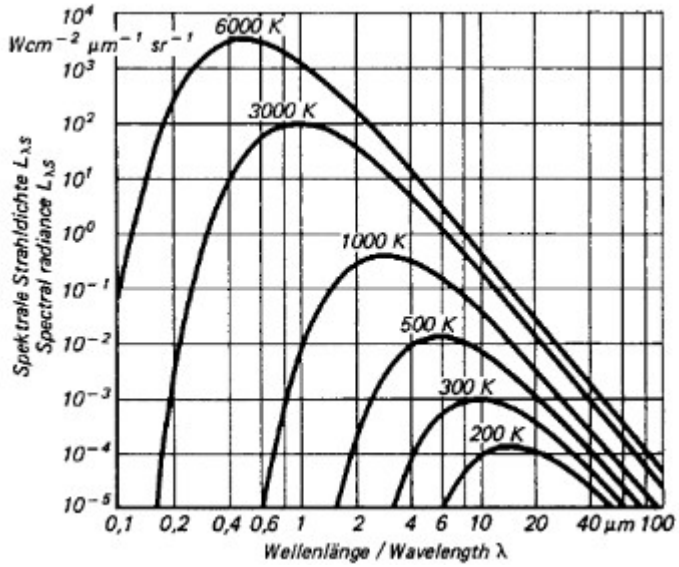


*Image 3: Radiation characteristics of a blackbody in relation to its temperature*

In image 3 the typical radiation of a blackbody at different temperatures is shown.

The amount of energy transmitted in a specific part of the spectrum depends on the temperature (higher temperature equals more energy) and some parts of the spectrum contain much more energy than others for the same temperature. The peak intensity for a given temperature moves to smaller wavelengths as the temperature increases.

Depending on the measured part of the spectrum the temperature differences are easier to detect. For instance, at the point at 40µm the differences between the temperatures are very small, at 2µm they are much bigger. However at 0.2µm it would not be possible to detect the lower temperatures at all.

So to take this into account a sensor is built for a specific wavelength range which also determines its temperature range.

### 2.1.4 Emissivity ε

The ratio between the real emitted power and that of a blackbody is known as emissivity ε. It is a value between 0 and 1 (1 being the ideal blackbody). Objects with an emissivity of less than 1 are called gray bodies. Bodies whose emissivity is also dependent on temperature and wavelength are called non-gray bodies.

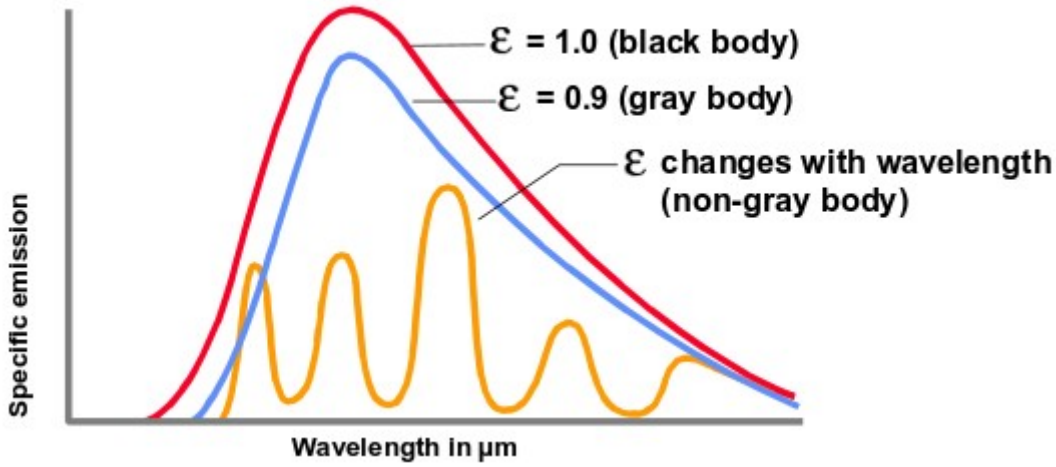


*Image 4: Specific emission at different emissivities*

## 2.2 Selecting a FIR sensor

The search for a FIR sensor needs to take the previous theoretical parts into account.

When searching for a sensor there are a lot of industrial designs available. These standalone sensors are already in a rugged package meant to be easily installed in any environment that meets their specifications. They contain the optics, sensor and the electronics needed to transmit the data.

*Image 5: CALEX Electronics Limited PyroCouple*

However the actual sensors that are used in those devices are often also available. And depending on the specifications additional optics are not required. This saves a lot of money as the sensing component alone is often less expensive by a factor of 10. Without additional optics there is also no need to do additional calibration and only need to take the emissivity of the target into account.

### 2.2.1 Available sensors

*Table 1: **Texas instruments: TMP007 and TMP006***

| | **TMP007** | **TMP006** |
|---|---|---|
| IR Sensor Accuracy (max) (+/- °C) | 3 | 3 |
| Local sensor Accuracy (max) (+/- °C) | 1 | 1 |
| Temp range (°C) | -20 to +125 | -20 to +125 |
| Supply voltage (min) (V) | 2.2 | 2.2 |
| Supply voltage (max) (V) | 5.5 | 3.6 |
| Interface | I2C / SMBus | I2C |
| Pin/Package | 8DSBGA | 8DSBGA |
| Integrated Math Engine | yes | no |
| Approx. Price(US$)(per 1000) | 1.90 | 1.50 |

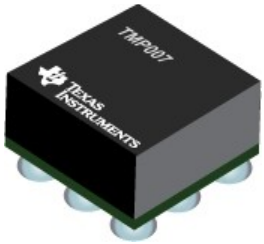


*Image 6: Texas Instruments TMP007*

Of these two sensors available from Texas Instruments the TMP007 seems like the best choice. It can do the required calculations to take emissivity into account.

*Table 2: **PIR Sensor Co LTD: PIR_D203B, PIR_D203S and PIR_D202X***

| | PIR_D203B | PIR_203S | PIR_D202X |
|---|---|---|---|
| Supply voltage(V) | 3-15V | 3-15V | 3-15V |
| Field of view x(°) | 110° | 138° | 113° |
| Field of view y(°) | 120° | 125° | 90° |
| Spectral response | 5-14µm | 5-14µm | 5-14µm |


*Image 7: PIR Sensor Co LTD PIR_D203B*

These sensors could be used to measure temperature but they are not designed with that in mind. They are used for motion detection so their accuracy is not as important. Additionally their field of view is too wide and all compensation and calculations would need to happen on the microcontroller.

*Table 3: **Measurement specialties TM: TS105-10L5 5mm and TS305-10C50***

| | TS105-10L5 5mm | TS305-10C50 |
|---|---|---|
| Operating Ambient Temperature (°C) | -20 to +85 | -20 to +85 |
| Package | TO-5 | TO-5 |
| Voltage Response (mV) | 0.9 ± 0.25 | 6.5 ± 1.9 |
| Field of View | 10° | 80° |
| Heat range | unspecified | unspecified |

*Image 8: Measurement Specialties TS105-10L5*

The TS105-10L5 has a good field of view that might be suitable for our application. However it is a “dumb” device consisting of a thermopile that will measure the radiated heat and an NPN resistor to measure the internal heat. All calibrations and calculations will have to be done externally. The actual temperature that can be measured is not mentioned in the datasheet however it is suggested to be at least from -10 to 100°C

*Table 4: **Amphenol Advanced Sensors: ZTP-115ML IR Module and ZTP-315MIH IR Module***

| | ZTP-115ML | ZTP-115M | ZTP-315MIH |
|---|---|---|---|
| Supply voltage (V) | 5 | 5 | 5 |
| Output mode | I²C | Voltage output | Voltage output |
| Temperature range (°C) | -20 ~ +100 | -40 ~ +145 | +20 ~ +380 |
| Field of view | ±5° | ±5° | ±9° |

*Image 9: Amphenol Advanced sensors ZTP-315MIH*

These modules have the benefit of being calibrated and most calculations are done on the module. To use them all you need to do is read the value. Though you still need to take emissivity into account yourself.

*Table 5: **Melexis: MLX90614, MLX90615, MLX90616***

| | MLX90614 | MLX90615 | MLX90616 |
|---|---|---|---|
| Sensor temperature (°C) | -40 to +125 | -40 to +85 | -40 to +85 |
| Object temperature (°C) | -70 to +380 | -40 to +115 | -70 to +1030 |
| Output mode | SMBus / PWM | SMBus | SMBus |
| Supply voltage | A=5V B=3V | 3V | 3V |
| Field of view | 5° to 35° | 100° or 80° | Needs external optics |
| Package | TO-39 | TO-46 | TO-39 |

*Image 10: Melexis MLX90614*

Each of these melexis sensors has different types with slight variations in the specifications, mostly regarding the field of view and the operating voltage.

All these sensors have internal processing to calculate the correct temperature and can take emissivity into account.

The MLX90614 series is widely used and comes in versions that are very suited to our needs

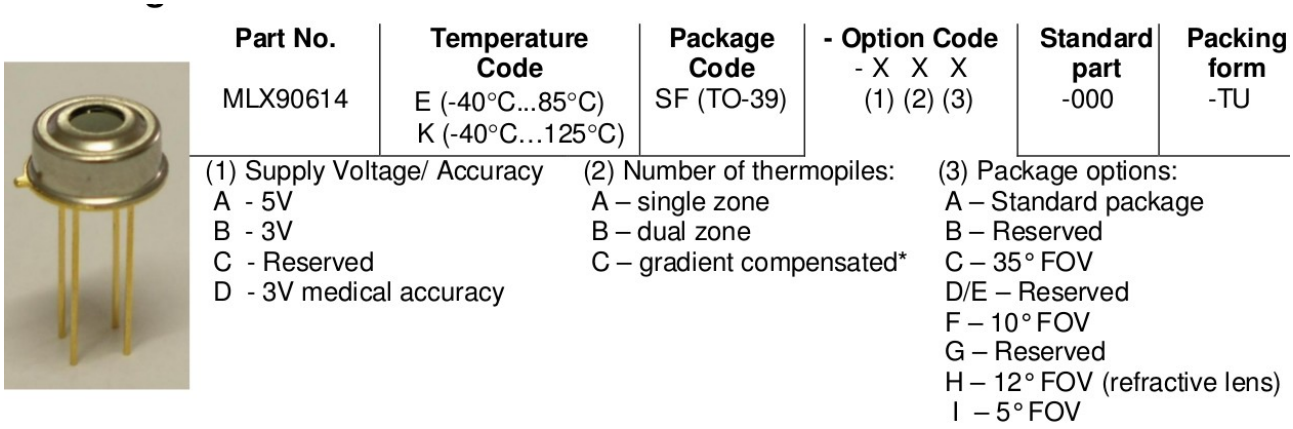


*Image 11: MLX90614 different types*

### 2.2.2 Overview and prices

The following will be a list of sensors that have a narrow field of view to be able to use at a distance. Those that are not easily available at distributors have been left out.

| Sensor | Field of View | Temperature range | Math Engine | Amount in stock | Price | Distributor |
|---|---|---|---|---|---|---|
| TMP007 | unknown | -20 to 125°C | yes | 404 | 6.98 | Digi-Key |
| TMP006 | unknown | -20 to 125°C | no | 18331 | 4.78 | Digi-Key |
| TS105-10L5 | 10° | -10 to 100°C | no | Request a quote | | |
| ZTP-115M | 5° | -40 to 145°C | no | 978 | 12.53 | Digi-Key |
| MLX90614ESF-ACF | 10° | -70° to +380° | yes | 1284 | 25.79 | Digi-Key |
| MLX90614ESF-BCI | 5° | -70 to +380° | yes | 906 | 35.83 | Digi-Key |

*Table 6: IR temperature overview and prices*

### 2.2.3 Conclusion

The MLX90614 is the most suited, it is available with a field of view of 10° or 5° allowing measurements from a large enough distance. It is also the only one with such a large temperature range.

All the MLX90614 versions have temperature and emissivity compensation build in so we just need to set the correct emissivity in the sensor and can start reading temperatures with fairly high accuracy.

The downside is that these chips are the most expensive but considering the potentially high temperature and the safe distance we need to maintain from a transformer it is the most logical choice.

An alternative would be to use one of the cheaper devices with a wide field of view and add additional optics to them. However those optics would themselves increase the cost and the sensor would need to be re-calibrated for those specific optics.

## 2.3 Acquiring a FIR sensor

Because of price constraints I reached out to Melexis about potential samples. Despite not having a sample program they ware kind enough to send some samples to the university free of charge.

# 3 Ozone sensor

## 3.1 Theory

For this project the absolute accuracy is not as important, only a rapid(relatively) increases in ozone needs to be detected which would mark a corona defect.

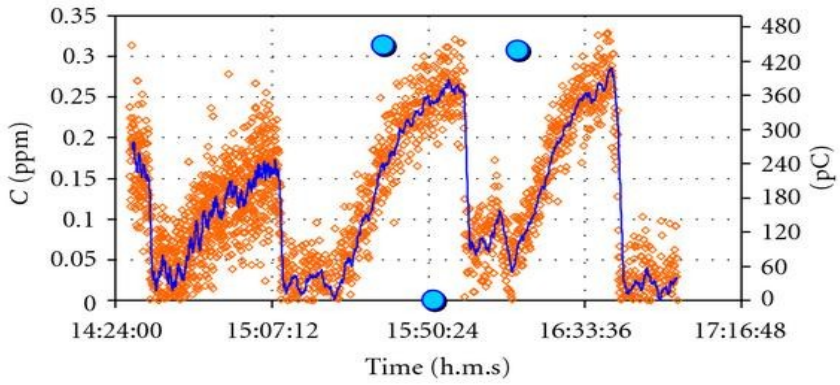


*Image 12: Ozone sensor response versus time measured in presence of the corona defect. The right vertical axis reports predischarge aplitudes (pC) by the standard PD electrical method.*

This graph comes from measurements inside a MV switchboard(with a custom fiber-optic sensor), our sensor however will be outside the switchboard. Testing is needed to determine how effective the measurements are or if the distance makes useful measurements impossible.

## 3.2 Selecting an ozone sensor

### 3.3.1 Available sensors

*Table 7: **MQ-131***

| | |
|---|---|
| Model | MQ131 |
| Sensor Type | Semiconductor |
| Standard Encapsulation | Plastic cap |
| Target Gas | Ozone |
| Detection range | 10 ~ 1000 ppb Ozone or 10 ~ 2000 ppb Ozone |

*Image 13: MQ-131 Gas Sensor*

This sensor needs to be heated up to function, the preheat time for optimal performance is very long (48 hours). Although you can measure things before the time is up, the accuracy will be degraded.

It is also important to take humidity and temperature into account for the best performance.

There appear to be different versions of this part with different datasheets available. Though the websites never mention which part it is or how to tell the difference.

This sensor costs $12.90 per single unit at futurlec.com.

*Table 8: **SGX Sensortech MiCS-2614 and MiCS-2610***

| Characteristic | Symbol | MiCS-2614 | | | MiCS-2610 | | | |
|---|---|---|---|---|---|---|---|---|
| | | typical | min | max | typical | min | max | Unit |
| Sensing resistance in air | $R_0$ | 11 | 3 | 60 | 11 | 3 | 60 | kΩ |
| Typical detection range | FS | | 10 | 1000 | | 10 | 1000 | ppb |
| Sensitivity factor | $S_R$ | 2 | 1.5 | 4 | 2 | 1.5 | 4 | - |
| Heating power | $P_H$ | 80 | 66 | 95 | 80 | 66 | 95 | mW |

*Image 14: MiCS-2614 Gas Sensor*

The MiCS-2610 and -2614 are almost identical in specification, the only difference is their packaging. The older -2610 however is no longer recommended and is no longer manufactured.

This sensor has a very low preheat time.

This sensor costs $8.10 per single unit at the SGX sensortech online store.

### 3.2.2 Problems finding a sensor

#### *Limited availability*

There are of course pre-calibrated hand held devices available on the market to detect ozone concentrations.

These are however far too expensive and only the sensors they use are interesting for this project. These sensors are not widely available, I have only found 2 possible sensors (and an older version of one of them) suitable for our needs.

There are more that have been ignored for being way too expensive.


*Image 15: A-22 Rugged hand held ozone sensor*

#### *Careful handling*

Care needs to be taken when soldering these sensors. Avoid wave soldering during manufacturing and even flux vapors could degrade the sensors.

During installing they should also be kept away from cigarette smoke or any corrosive vapor.

#### *Limited lifespan*

Some of these sensors only guarantee a lifespan of >18months, some do not specify it at all.

#### *High shipping costs*

Besides availability, the shipping costs are often way higher than the actual product. For a single $8.10 part the recommended reseller asked about $120 for shipping.

### 3.2.3 Decision

The MiCS-2614 is the newest device and the cheapest.
It is however harder to solder than the MQ131 and will require a SMD reword station or a reflow oven.

## 3.3 Acquiring an ozone sensor

Due to the very high shipping costs and no available samples we have decided to **not** buy the ozone sensor.

Because the sensor is a fairly dumb device (a variable resistor) it is simulated with a potentiometer. However the footprint is included on the PCB for future use.

# 4 Humidity sensor

## 4.1 Theory

Humidity refers to the the water vapor content in air or other gases. There are several units to express humidity. The 3 commonly used terms are absolute humidity, dew point, and relative humidity(RH).

There are 3 main types of humidity sensors:
- Capacitive Humidity Sensors
- Resistive Humidity Sensors
- Thermal Conductivity Humidity Sensors

Of these the capacitive relative humidity sensor is the most used. They are accurate, dependable and pretty cheap to produce.

## 4.2 Selecting a humidity sensor

### 4.2.1 Available sensors

Unlike the previous sensors there is a large selection of humidity sensors. Because of this the search is focused to those that cost less than 15 euro (for a single sensor) and are still being produced.

| Sensor | Supply voltage (V) | Accuracy (%RH) | Package | Interface | Price (Euro) | Supplier |
|---|---|---|---|---|---|---|
| Texas Instruments HDC1008 | 3~5 | ±4 | BGA | I²C | 4.69 | Digi-Key |
| Silicon Labs SI7013-A20 | 1.9 - 3.6 | ±3 | DFN | I²C | 6.07 | Digi-Key |
| Silicon Labs SI7020-A20 | 1.9 – 3.6 | ±4 | DFN | I²C | 4.78 | Digi-Key |
| Measurement Specialties HTU21D | 1.5 ~ 3.6 | ±2 | DFN | I²C | 11.59 | Digi-Key |
| Honeywell Sensing and Control HIH6130 | 2.3 – 5.5 | ±4 | SMD | I²C/SPI | 13.03 | Digi-Key |
| Honeywell Sensing and Control HIH7131 | 2.3 – 5.5 | ±3 | SMD | SPI | 8.65 | Digi-Key |

*Table 9: Humidity sensors*

### 4.2.2 Decision

The Silicon Labs SI7020-A20 is the best option. It is the cheapest one available (that is still in production) and would be easier to solder than a BGA package.


*Image 16: Silicon Labs SI7020-A20 Humidity Sensor*

However if there is no hot air rework station available soldering can be quite hard. A good alternative would be the HIH6130 or HIH6131.

## 4.3 Acquiring a humidity sensor

Silicon Labs has an excellent sample program so getting a humidity sensor was no problem at all.

# PART 2: Hardware design

# 1 CAD software

## 1.1 Available options

### 1.1.1 Altium Designer

This is the software used in the Belgian university (University College Leuven Limburg) and the one I'm most familiar with. It has a ton of features and is great for professional designs.

The downside is the huge license fee.


*Image 17: Altium logo*

### 1.1.2 Eagle CAD

This is the software used at the Brno University of Technology, it also has a lot of features. And there is a free version available which limits how big your PCB can be but is perfectly usable for most projects.


*Image 18: Eagle logo*

### 1.1.3 KiCAD

This is an opensource electronics CAD package. I have begone learning to use this software for when my altium license expires.

It can do everything required without any limits,. However the interface is not very user-friendly and takes some getting used to.


*Image 19: KiCAD logo*

## 1.2 Decision

I decided to go with Altium designer because I am most familiar with it and I wanted to more deeply explore some of its features.

# 2 Design requirements

## 2.1 MSP430

The use of an MSP430 is one of the design requirements.

The MSP430 is a mixed-signal microcontroller family from Texas Instruments. It is built around a 16-bit CPU and is designed for low cost and low power consumption embedded applications.

An MSP-EXP430F5438 Experimenter Board was used during the design phase to get familiar with the platform. This board contained an MSP430F5438A microcontroller.

The same MSP430 is used in the prototype to be able to reuse a lot of software.

This controller contains:
- 3 16-bit Timers
- 4 Universal Serial Communication Interfaces
- 12-bit Analog-to-Digital Converter (ADC)
- 32-bit hardware Multiplier
- Serial Onboard Programming
- Basic timer with Real-Time Clock Feature

*Image 20: MSP430F5438A*

The microcontroller was generously provided through the Texas Instruments Sample program.

## 2.2 Modbus

An important criteria is including Modbus capabilities to the hardware.
This meant including an UART to RS485 chip on the PCB so the Modbus functionality could be programmed later.

The SN65HVD78 from Texas Instruments was used.
This 3.3V chip has robust ESD protection and a 50Mbps data rate.

*Image 21: SN65HVD78*

The chip was also provided through the Texas Instruments Sample program.

## 2.3 PCB

The PCB was manufactured inside the university. This means there are more constraints than using a professionally manufactured PCB.

The university can do 2-layer, no through-hole plating, no silk screen and no solder mask.

Besides those constraints the process has the following limits:

1) minimum trace width: 0.25mm
2) minimum gap width: 0.25mm
3) minimum distance from polygon: 0.6mm
4) minimum hole diameter: 0.6mm

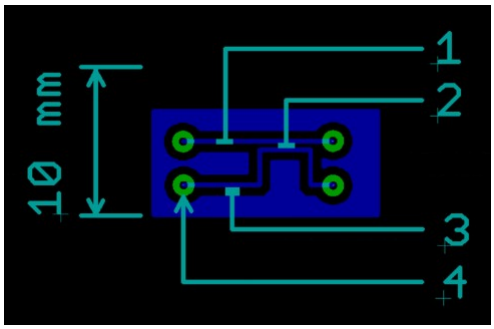

*Image 22: PCB constraints*

# 3 Schematic

The schematic is split into 4 separate parts:

- Power: selecting power source, generating stable 5V and 3.3V
- Controller: connections to microcontroller and microcontroller power filers
- Communication: conversion to 5V logic and UART to RS485
- Sensors: humidity, temperature and ozone sensors

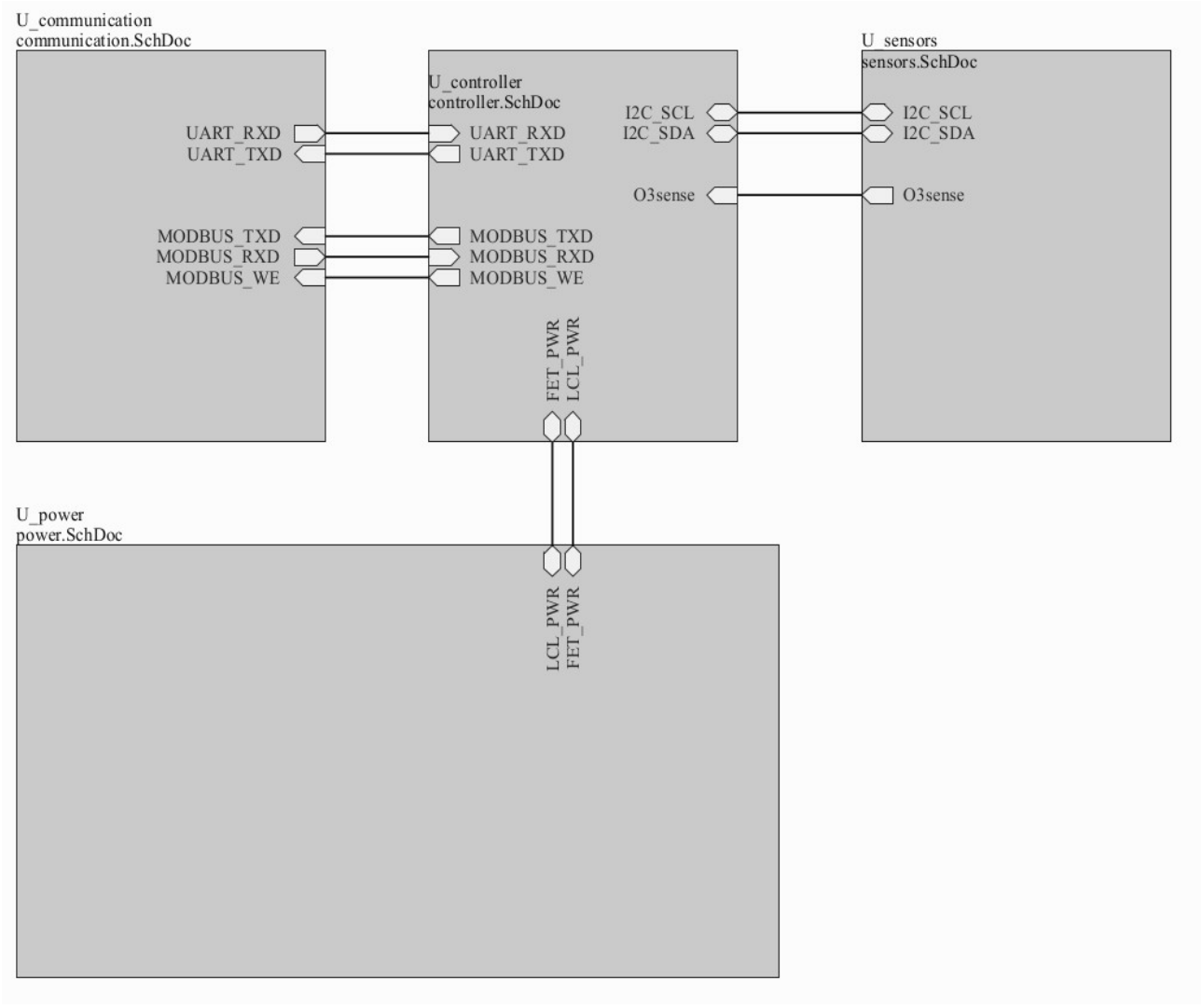


*Image 23: Schematic Layout*

# 3.1 Power

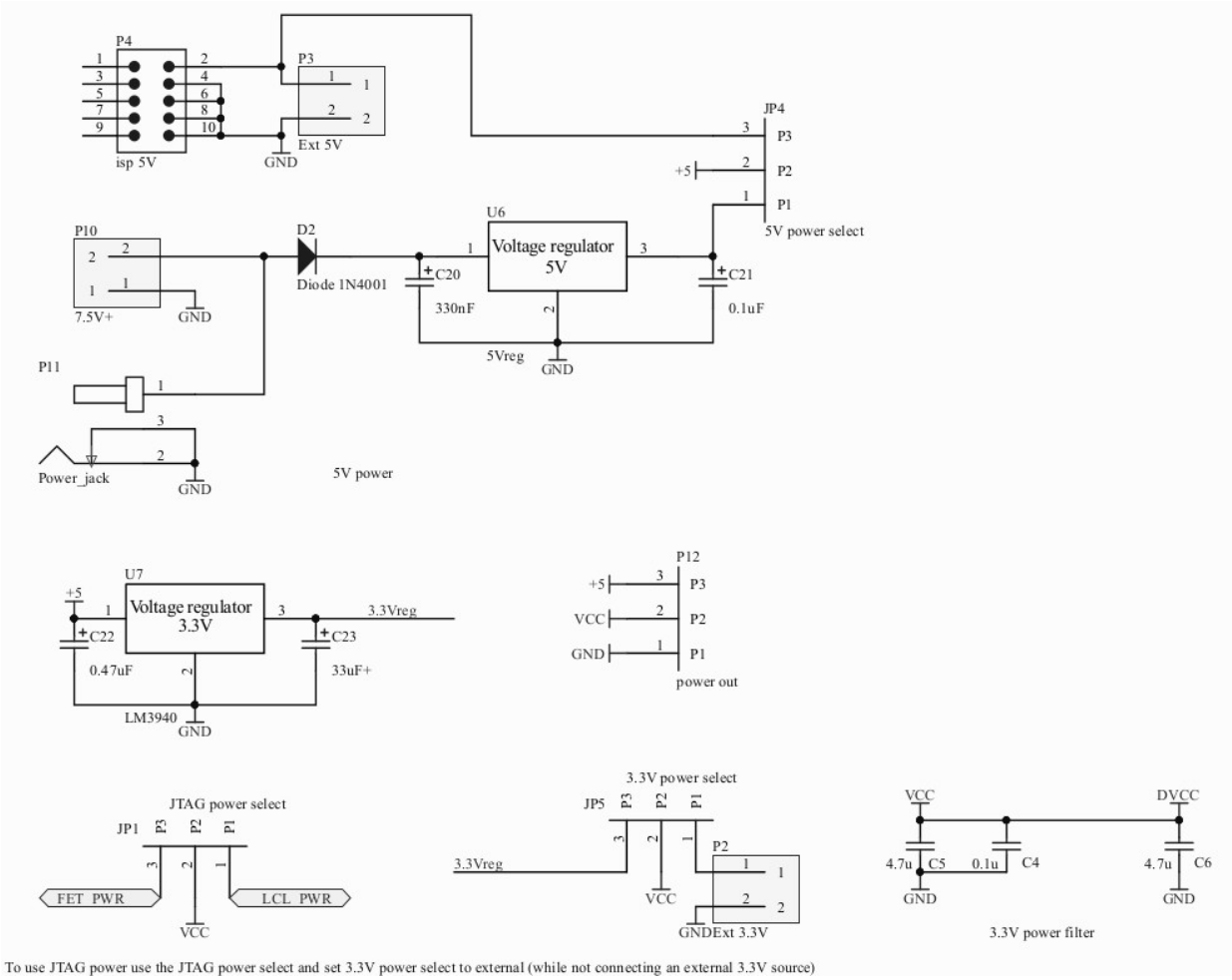


*Image 24: Schematic Power*

## 3.1.1 Connections

There are several ways to power the board:

| | |
|---|---|
| External 5V:<br>- ISP programmer header (P4)<br>- Screw connector (P3) | External >7.5V:<br>- Screw connector (P10)<br>- Power jack (P11) |
| External 3.3V:<br>- Screw connector (P2) | By placing jumpers the 5V and 3.3V can be generated by the linear voltage regulators. |

## 3.1.2 Linear Voltage Regulators

There are 2 linear voltage regulators, one to go from >7.5V to 5V. And one to go from 5V to 3.3V.

## 3.1.3 Filter

A filter is placed on the 3.3V line with an additional filter cap for DVCC which powers the microcontroller.

## 3.2 Controller

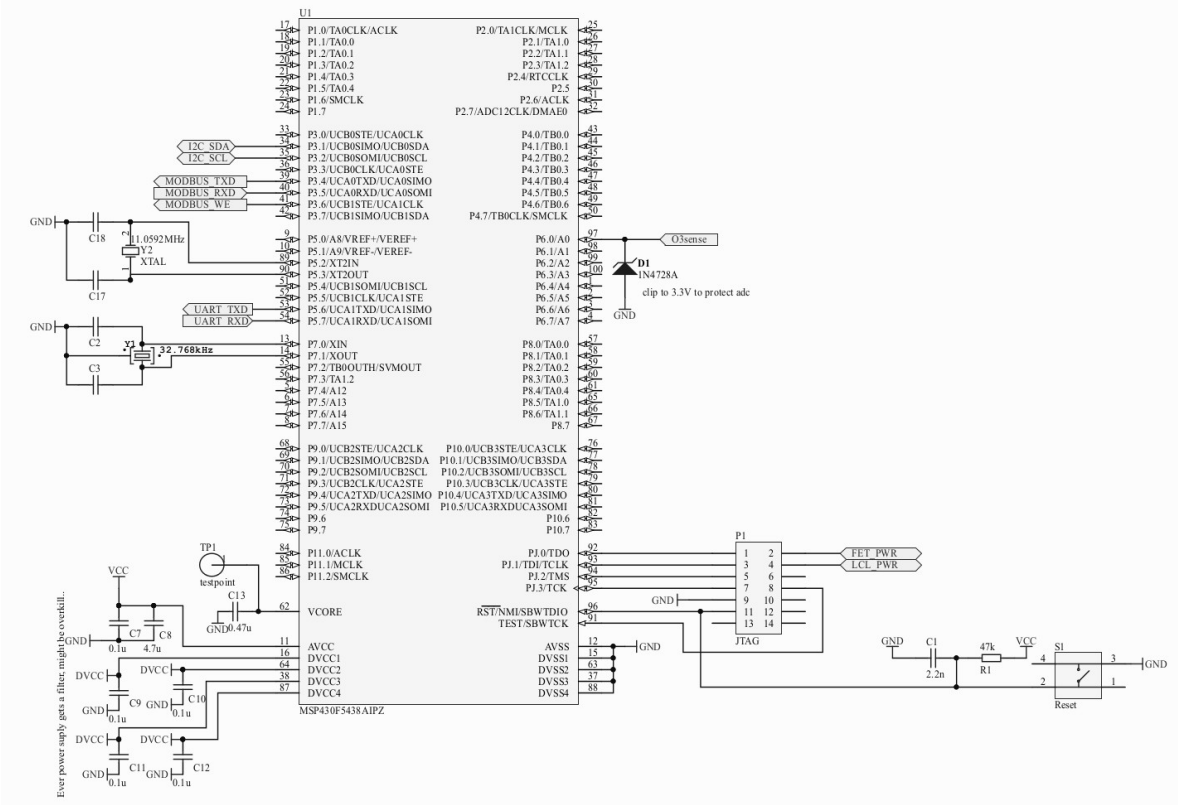


*Image 25: Schematic Controller*

### 3.2.1 Pins

Most pins are not used in this design. The controller was chosen for ease of use during development and available free samples. In a production version it should be replaced with a smaller controller.

### 3.2.2 JTAG

There is a JTAG header to program the controller and a reset button.

### 3.2.3 Crystals

A 32.768kHz watch crystal is used for XTAL1, this will allow the controller to work on very low power (but also very slowly).

Additionally a 11.0592MHz crystal is used for XTAL2, this sets the speed for the peripheral such as UART(modbus), SPI, clocks and more. 11.0592MHz is easily dividable to clock speeds of standard baud rates and thus makes the communication much more stable.

### 3.2.4 Filters

There are filters for the controllers power. During layout these are placed as close as possible.

### 3.2.5 Zener

A 3.3V zener is used to make sure the analog input does not go too high.

## 3.3 Communication

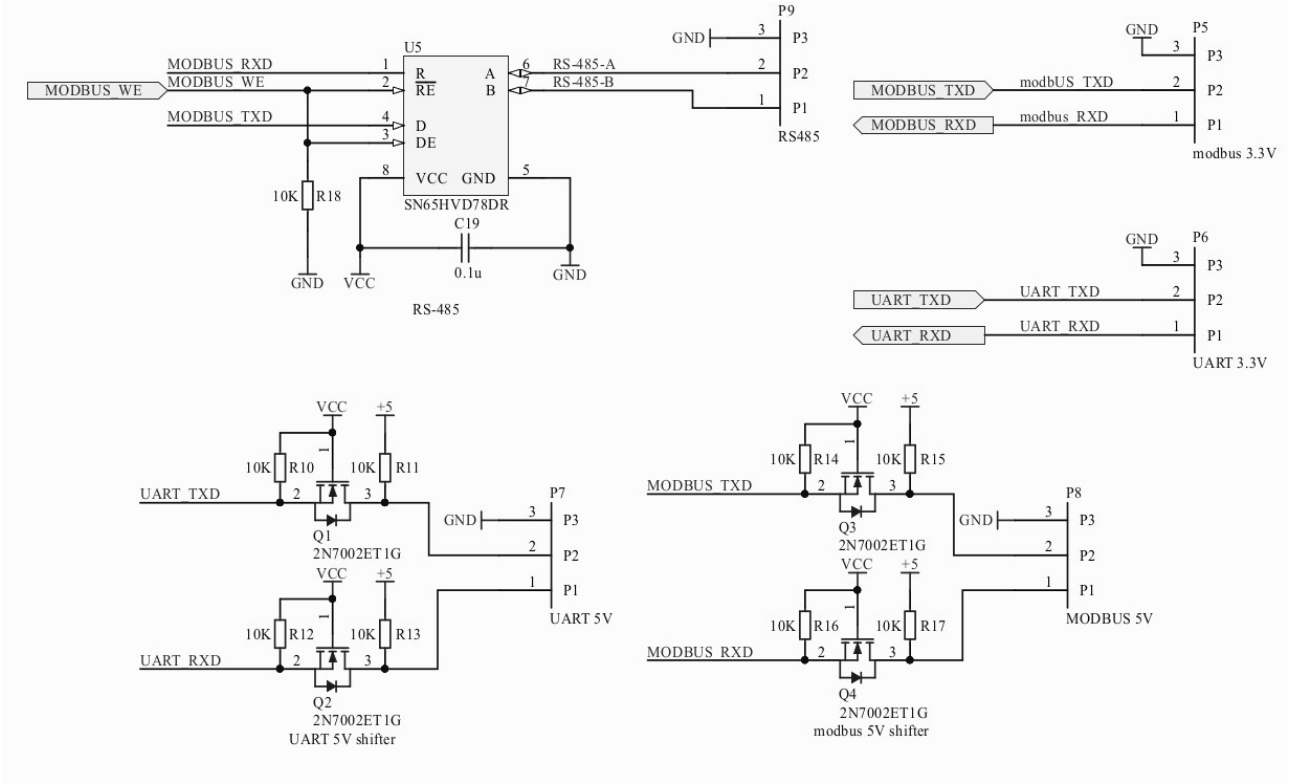


*Image 26: Schema Communication*

### 3.3.1 Modbus

Modbus uses RS485. An SN65HVD78DR is included to covert from UART to RS485. It is half-duplex so the MODBUS_WE pin is used to select reading or writing.

### 3.3.2 3.3V to 5V

Because the controller might want to communicate with 5V chips, it also contains a 3.3V to 5V conversion. This conversion can work on input and output pins.

#### *Example if UART_TXD is an output*

If UART_TXD is high, there is 0V over R10, the FET does not conduct and the 5V pull-up (R11) makes the output high.

If UART_TXD is low, there is a voltage over R10 and the fet conducts, pulling the 5V pull-up down to ground.

#### *Example if UART_TXD is an input*

If the input is high, nothing happens: the signal on UART_TXD would also be high because it is pulled high by R10.

If the input is low, it will pull down R10 through the diode in the FET.

## 3.4 Sensors

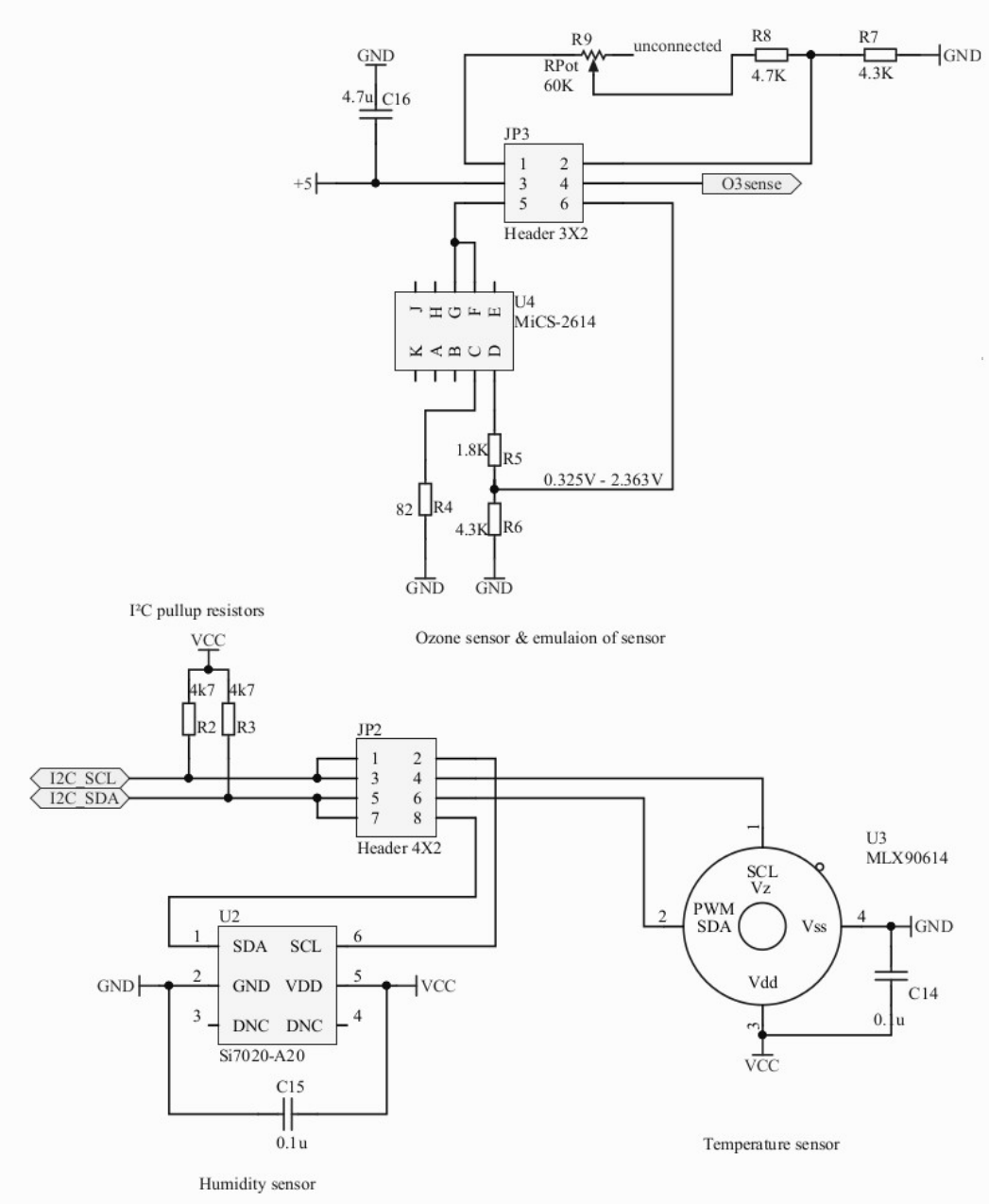


*Image 27: Schematic Sensors*

### 3.4.1 I²C pull-up

R2 and R3 are pull-up resistors for the SCL and SDA lines.

### 3.4.2 I²C Sensor selection

The humidity sensor (SI7020-A20) is a traditional I²C device, while the temperature sensor (MLX90614) is a SMBus device.

SMBus is mostly compatible with I²C but has some slight differences. The major difference is SMBus has a clock time-out of 35ms, while with I2C you can stop the bus for however long you want. To aid in debugging you can remove either device from the I²C bus by using jumpers JP2.

### 3.4.3 Ozone simulation

Because of the price of the ozone sensor (MiCS-2614) there is a simulation of the ozone device using a potentiometer.

The user can select whether to use the ozone sensor (if one is available) or the simulation with jumper JP3.

The output voltage on O3sense will be roughly the same in both cases.

# 4 PCB

## 4.1 Design tricks

Because it was not clear which parts would be available for the prototype, the design is made to be as flexible as possible. With some forethought the same footprint can accommodate several different parts and it no longer matters which parts are more readily available.

### 4.1.1 Voltage Regulator

A linear voltage regulator is simple to use and usually has 3 leads:

- Unregulated voltage input
- Regulated voltage output
- Ground

There are several different regulators available, with the order of the leads and the type of package varying from part to part. Notice in the picture how none of the pinouts match.

By adding a 4th and 5th hole to the layout the footprint covers all 6 possible combinations of pins. And by making sure all the holes are at least 1mm big, it can fit both TO-220 and TO-92 packages.

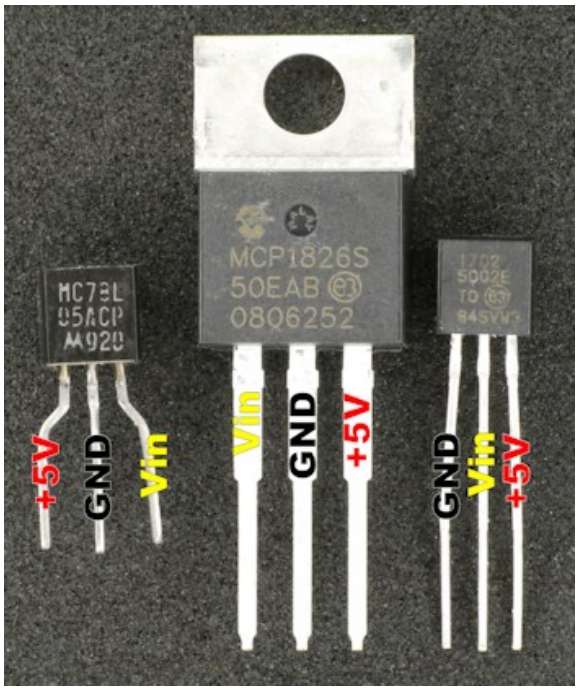


*Image 28: Voltage regulator pinouts: 78L05, MCP1826, MCP1702-5002E*

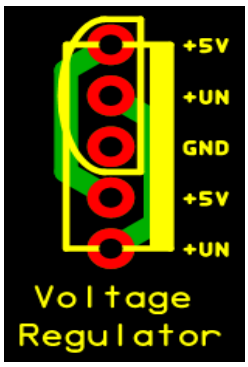


*Image 29: Flexible voltage regulator PCB layout*

The changed layout permits the following pinouts:

- +5, +UN, GND (top three holes)
- GND, +UN, +5 (installed backwards in top three holes)
- +UN, GND, +5 (middle three holes)
- +5, GND, +UN (installed backwards in the middle three holes)
- GND, +5, +UN (bottom three holes)
- +UN, +5, GND (installed backwards in the bottom three holes)

### 4.1.2 Trimmer potentiometers

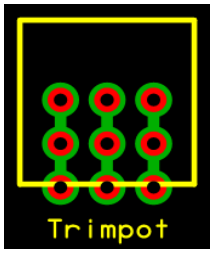


*Image 30: Potentiometer multi-compatible PCB layout*

Small potentiometers are available in a lot of different packages.
Most of them have 2.56mm spaced leads but their position can vary a lot.

By laying out a 3x3 grid, the footprint is compatible with all the pinouts.

This pattern could be expanded even more to take variations in the pin number of the wiper into account.
However since the middle pin is the wiper in most components this design is sufficient.

### 4.1.3 Through-hole capacitors


*Image 31: Capacitor pattern with multiple lead spacings*

With uncertainty about the available linear power supply, there is naturally also uncertainty about the capacitors required for them. Going with through-hole capacitors (unlike the SMD ones used in the rest of the board) should allow for easy to find large value capacitors.

Because they also come in different sizes the capacitors also have an extended footprint.

## 4.2 Potential problems

### 4.2.1 Trace width / clearance constraints

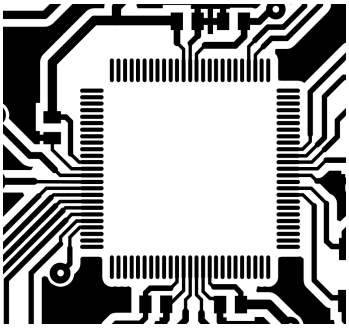

*Image 32: Microcontroller pin spacing*

The constraints of the manufacturing process specify how wide traces can be and how much clearance they need to guarantee it can be produced.

The pin spacing for the microcontroller actually violates these constraints, but those in charge of production allow it for pins on a device as long as you separate the traces as soon as possible.

The image illustrates how narrow paths become wider as soon as possible.

### 4.2.2 No through-hole plating

Because there is no through-hole plating there is no guarantee that a component is connected on both sides unless if you manually solder both sides (which isn't always possible).

To prevent any problems all traces to and from a through-hole component are on the side where they would be soldered.

Additionally all vias have enough clearance so they can be manually created by putting some wire in the hole and soldering both ends.

### 4.2.3 Component placement

All sensors and connectors need to be facing the same side of the board. The IR temperature sensor needs to be pointing towards the transformer and both humidity and ozone sensors need enough airflow to be able to measure correctly.

Some components such as the ozone sensor, humidity sensor, microcontroller, and linear voltage regulators generate heat. This heat could interfere with the IR temperature readings so those devices are placed away from the IR temperature sensor.

### 4.2.4 Soldering

Most components on the board are very small and there is no solder mask. Hand soldering is possible (as the prototype proves) but can be hard and frustrating at times.

When soldering you need to take care to protect all the sensors.
They are sensitive and should not get any flux vapors in the sensor openings.

### 4.2.5 No silk screen

There is no silk screen so important markings are placed in the copper itself.

# 5 Result

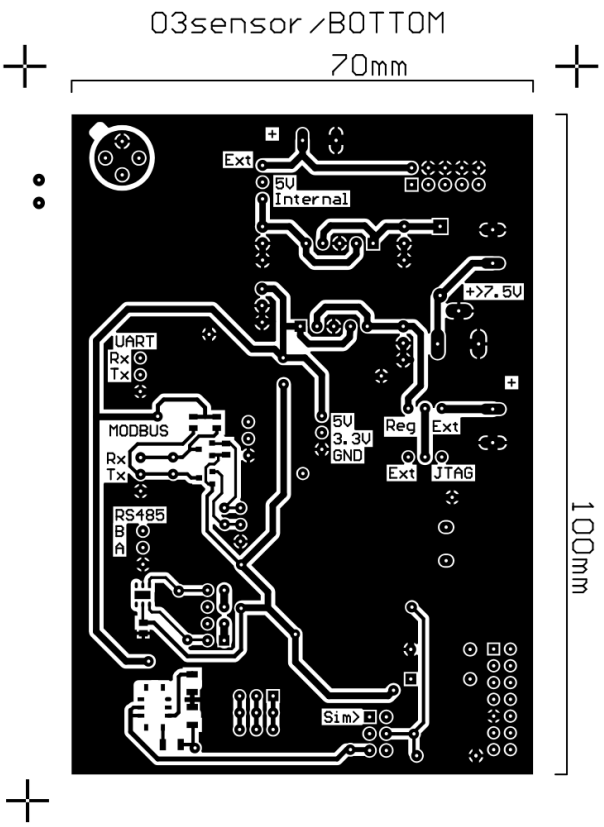


*Image 34: PCB bottom*

The end result is a compact 10cm by 7cm prototype board.

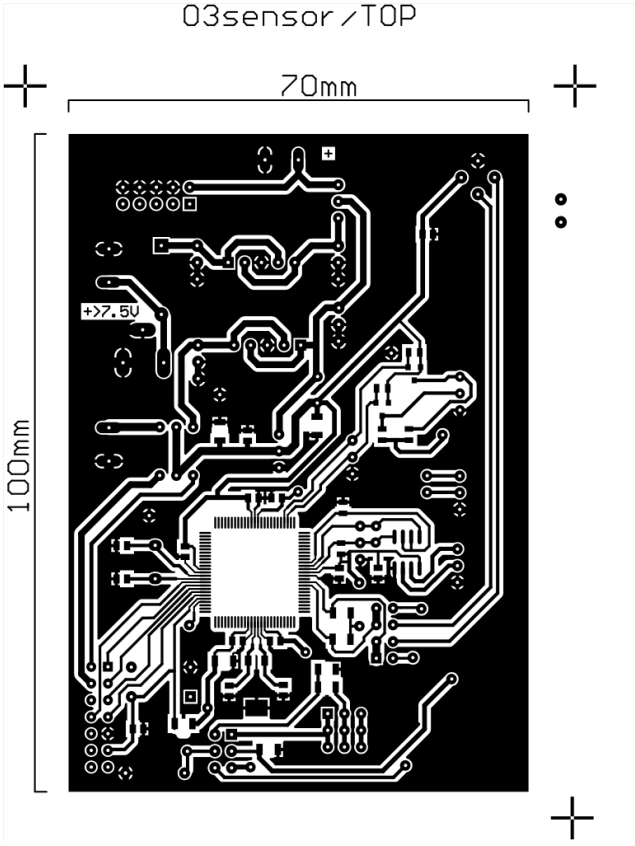


*Image 33: PCB top*

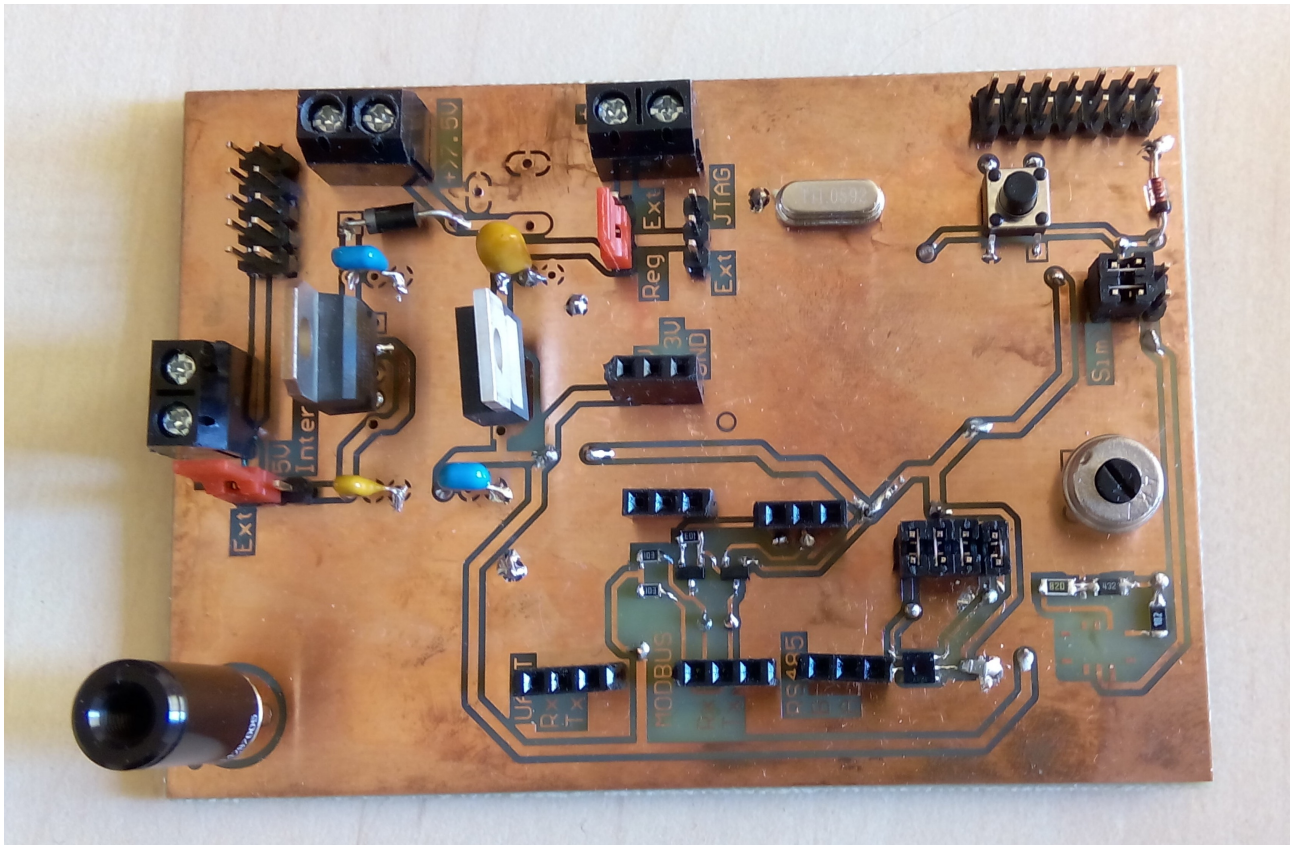

*Image 35: Finished PCB prototype*

# PART 3: Software

# 1 IDE Software

## 1.1 Available options

### 1.1.1 Code Composer Studio

Code Composer Studio (CCS) is an integrated development environment (IDE) that supports Texas Instrument's Microcontroller and Embedded processor portfolio. It includes an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler, and many other features.

This IDE is based on the Eclipse IDE, but with all the Texas Instruments stuff pre-loaded, which saves a lot of setup time. (linux, mac and windows support)

### 1.1.2 IAR embedded workbench

IAR is another IDE suited for the MSP430 platform. However it is not free and requires a license.

### 1.1.3 CrossWorks for MSP430

Another non free IDE focussing on the MSP430 platform. (linux, mac and windows support)

### 1.1.4 Custom toolchain setup

There is of course no reason why you can not use your own favourite IDE to program an MSP430. This process involves a lot more work though, at the very least you are going to need the MSPGCC compiler.

## 1.2 Decision

I went with Code Composer Studio. It is offered by the manufacturer of the microcontroller itself and saves a lot of trouble. It has all the features required and full linux suport.

## 1.3 Initial setup

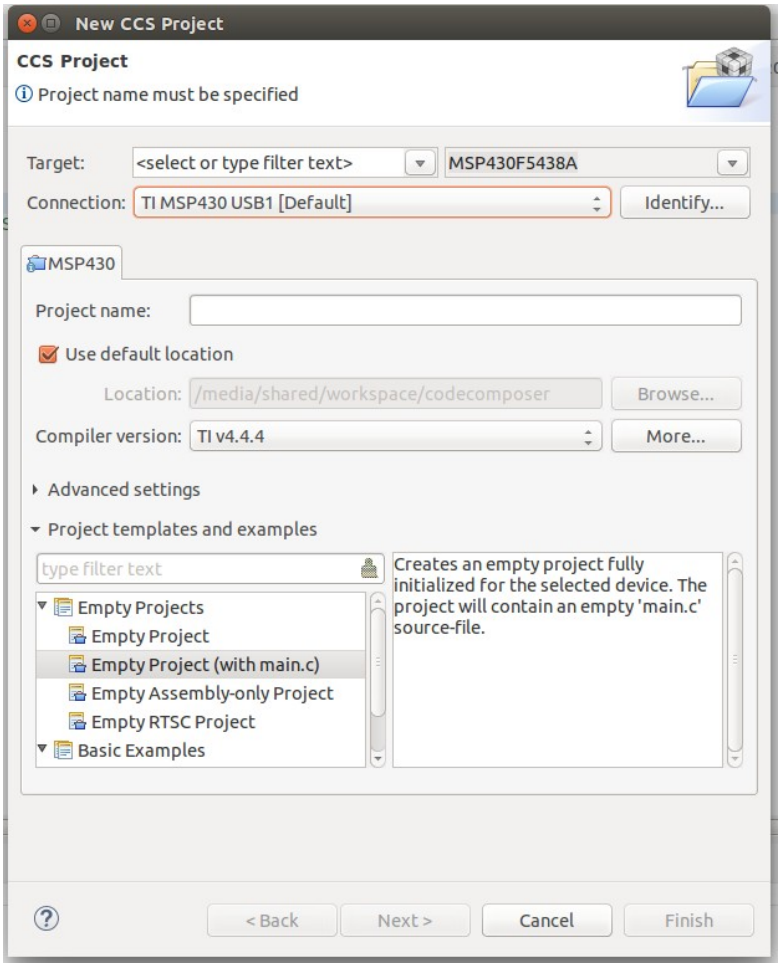


*Image 36: Code Composer Studio: new project*

To start a new project simply go to:

File->new->CSS project

You will get a wizard to help set up your project (see Image).

Make sure to select the proper target (MSP430F5438A) and give the project a name. After that press finish and start programming.

## 1.4 Debug printf support

Using printf in Code Composer Studio will print to the debug interface. However, it requires some settings to get it to work properly. The guide at http://processors.wiki.ti.com/index.php/Printf_support_for_MSP430_CCSTUDIO_compiler goes through all the required steps.

# 2 Additional software

## 2.1 screen utility

The screen utility available on all linux distributions works as an easy to use serial client. This tool was used a lot for testing the communication and debugging.

## 2.2 Modpoll

Modpoll is a command line utility that allows you to test your modbus implementations.

# 3 Code

The code on the microcontroller is plain C, it is split up in several source and header files (available in the attachments):

- main.c
- I2C.c I2C.h
- uart.c uart.h
- clock.c clock.h
- MLX90614.c MLX90614.h
- SI7020-A20.c SI7020-A20.h
- ozone.c ozone.h
- modbus.c modbus.h

## 3.1 Previous efforts

Before the current code was written a lot of effort went into porting the FreeModbus library to this controller. This library is quite big though and a lot of changes ware required to get it to a working state so those efforts ware abandoned.

## 3.2 Setting the clock

All parts in the code rely on very specific timing, so the first step in the program is to set the different clocks.

### 3.2.1 Clock sources

- XT1CLK: External crystal, can be either low-frequency or high-frequency
- VLOCLK: Internal very low power, low frequency oscillator with 10kHz typical frequency
- REFOCLK: Internal, trimmed, low-frequency oscillator with 32768Hz typical frequency
- DCOCLK: Internal digitally-controlled oscillator (DCO)
- XT2CLK: External high-frequency oscillator

### 3.2.2 Clock signals

- ACLK: Auxiliary clock, the ACLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available XT2CLK
- MCLK: Master clock, MCLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available XT2CLK
- SMCLK: Subsystem master clock. SMCLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available XT2CLK

DCOCLKDIV is DCOCLK frequency divided by 1, 2, 4, 8, 16 or 23 within the FLL block.

### 3.2.3 Code

```
void init_clock(void) {
      P7SEL |= 0x03;                        // pin P3.0,1 use as clock XT1CLK
      P5SEL |= 0x0C;                        // pin P5.2,3 use as clock XT2CLK

      UCSCTL6 &= ~(XT1OFF + XT2OFF);        // set XT1 & XT2 on
      UCSCTL6 |= XCAP_3;                    // Internal load cap

      //loop until XT1, XT2 & DCO stabilizes
      do {
            UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + XT1HFOFFG + DCOFFG);
                                    // Clear XT2,XT1,DCO fault flags
            SFRIFG1 &= ~OFIFG;      //Clear fault flags

      }while (SFRIFG1&OFIFG);       //test oscillator fault flag

      UCSCTL6 &= ~XT2DRIVE1;  // Decrease XT2 Drive according to
                        // expected frequency
                        // in this case 11.0592MHz so setting should be 01b
      UCSCTL4 |= SELA_0 + SELS_5;   // Select SMCLK, ACLK source and DCO source
```

} (from clock.c)

P7SEL and P5SEL sets specific pins on the microcontroller to use their special features (in this case, allowing us to attach a crystal to them). Turn on the crystals and add internal load capacity using UCSCTL6. Then wait until the clock stabilizes, after that decrease the drive according to the expected frequency using UCSCTL6. Set SMCLK, ACLK and DCO source with UCSCTL4.

The end result is as followed:

ACLK = LFXT1 = 32768Hz

MCLK = default DCO = 32 x ACLK = 1048576Hz

SMCLK = HF XTAL = 11.0592MHz

## 3.3 Reading the sensors

The design includes 2 different kinds of sensors, an analog one (the ozone sensor) and 2 using I²C (temperature and humidity sensors). They are in both cases being read in a blocking way, meaning code stops being executed until the sensor data is read.

### 3.3.1 Analog sensor (ozone)

```
void init_ozone(void)
{
	P6SEL |= 0x01;                            // P6.0 ADC option select
	ADC12CTL0 = ADC12SHT0_2 + ADC12ON;         // Sampling time, ADC12 on
	ADC12CTL1 = ADC12SHP;
	//ADC12IE = 0x01;                           // Enable interrupt for A0
	ADC12CTL0 |= ADC12ENC;                              // Enable conversion
}

void read_ozone(void)
{
	ADC12CTL0 |= ADC12SC;                     // Start sampling/conversion
	while (!(ADC12IFG & BIT0));
	if (count > 2)
		count = 0;
	last[count] = ADC12MEM0;
	count++;
} (from ozone.c)
```

Pin 6.0 is set as an analog pin, and the sampling time is set.

To read an analog value, start the analog to digital conversion and wait until it is done. This piece of code stores the last 3 values so they can be averaged out.

### 3.3.2 I²C (IR temperature and humidity)

I²C is a little more complicated than reading an analog value. To read data from a slave the master needs to send a start, followed by the address, the command, another start, the address again (with read bit set) and only then does the slave start giving useful data.

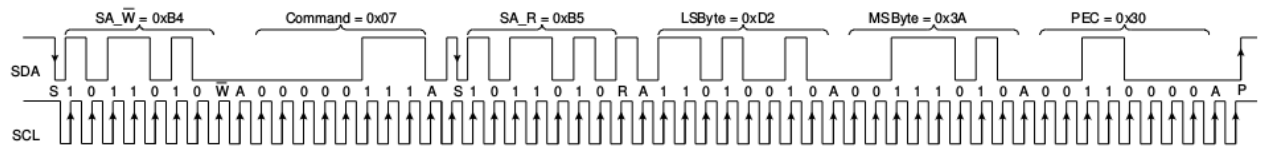


*Image 37: I2C Read word format*

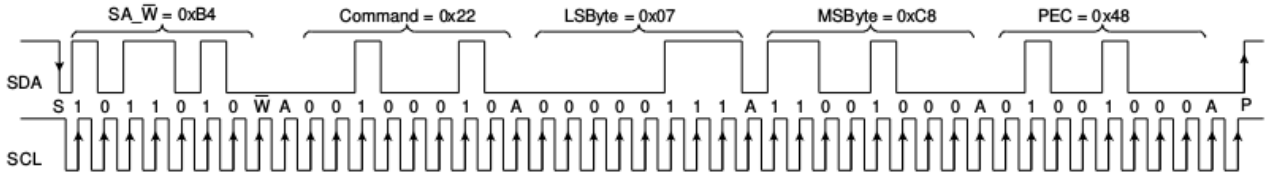


*Image 38: I2C Write word format*

### Start

```
void start_i2c(unsigned char Addr, unsigned char ReadWrite)
{
	P3SEL |= 0x06;                        // Assign I2C pins to USCI_B0
	while (UCB0CTL1 & UCTXSTP);           // ensure stop bit was sent
	UCB0CTL1 |= UCSWRST;                  // enable sw reset
	UCB0IFG &= ~UCTXIFG;                  //make sure flag isn't set
	UCB0CTL0 = UCMST + UCMODE_3 + UCSYNC;// I2C Master, synchronous mode
	UCB0CTL1 = UCSSEL_2 + UCSWRST;        // Use SMCLK
	UCB0BR0 = 120;          // fSCL = SMCLK/120, ~12MHz/120 = ~100kHz
	UCB0BR1 = 0;
	UCB0I2CSA = Addr;              //set slave addr
	UCB0CTL1 &= ~UCSWRST;          // Clear SW reset, resume operation
	if (ReadWrite)
		UCB0CTL1 |= UCTR + UCTXSTT;   // Send start bit
	else
	{
		UCB0CTL1 &= ~UCTR;
		UCB0CTL1 |= UCTXSTT;          // Send start bit
	}
} (from I2C.c)
```

### Transmit Data

```
void send_i2c(unsigned char value)
{
	while (!(UCB0IFG & UCTXIFG ));
	UCB0TXBUF = value;
} (from I2C.c)
```

Wait until transmit buffer is empty, then add the data to the buffer.

### Restart

```
void restart_i2c(unsigned char ReadWrite)
{
	if (ReadWrite)
		UCB0CTL1 |= UCTR + UCTXSTT;   // Send start bit
	else
	{
		UCB0CTL1 &= ~UCTR;
		UCB0CTL1 |= UCTXSTT;    // Send start bit
	}
	UCB0IFG &= ~UCTXIFG;    //make sure flag isn't set
} (from I2C.c)
```

### Receive Data

```
unsigned char read_i2c()
{
	unsigned char retval = 0;
	while (!(UCB0IFG & UCRXIFG));       // wait until RX buffer is full
	retval = UCB0RXBUF;     // Read RX buffer
	UCB0IFG &= ~UCRXIFG;    // make sure interupt flag is no longer set
	return retval;
} (from I2C.c)
```

### Stop

```
void stop_i2c(void)
{
      UCB0CTL1 |= UCTXSTP;    //I2C stop
      UCB0IFG &= ~UCTXIFG;    //make sure flag isn't set
} (from I2C.c)
```

### Full read command

```
unsigned int MLX_readTemp(void)
{
      unsigned char temp[3];
      start_i2c(MLX90614_I2C_ADDRESS, I2C_WRITE); //I2C start
      if (nack_i2c())   //check for NACK from slave
      {
            abort_i2c(); //I2C stop
            return 0;
      }
      __delay_cycles(100);
      send_i2c(MLX90614_TOBJ1); //send the command
      if (nack_i2c()) //check for NACK from slave
      {
            abort_i2c(); //I2C stop
            return 0;
      }
      restart_i2c(I2C_READ); //send start again
      if (nack_i2c()) //I2C check NACK from slave
      {
            abort_i2c(); //I2C stop
            return 0;
      }
      temp[0] = read_i2c(); //I2C read the data
      temp[1] = read_i2c();
      temp[2] = read_i2c(); //ignoring this checksum..
      stop_i2c(); //I2C stop
      objectTemp = ( ((unsigned int) temp[1]) << 8 ) | ( (unsigned int)
temp[0] );

      return objectTemp;
} (from MLX90614.c)
```

## 3.4 UART

### *Initialize*

```
void init_uart(void) {

      P3SEL |= 0x30;                // set P3.4,5 to USCI_A1 mode
      P3DIR |= 0x40;                //set P3.6 to output (Write enable pin)
      write_enable(0);

    UCA0CTL1 |= UCSWRST;            // **put state machine in reset**
    UCA0CTL1 |= UCSSEL_2;           //CLK = SMCLK

            //SMCLK = 11.0592MHz
            //devision factor N= fBRcLK/Baudrate
            // N = 11059200Hz / 115200Baud = 96 exactly!
            //oversampling baud rate UCBRx=INT(N/16) = 6

    UCA0BR0 = 6;
    UCA0BR1 = 0;
    UCA0MCTL = UCBRS_0 + UCBRF_0 + UCOS16;
                //UCBRS = 0, UCBRS=0, oversampling enabled
                //no modulation required, our xtal is awesome
    UCA0CTL1 &= ~UCSWRST;     // **enable USCI state machine **
    UCA0IE |= UCRXIE;
    __enable_interrupt();
} (from uart.c)
```

### *Read/write*

```
// Echo back RXed character, confirm TX buffer is ready first
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=USCI_A0_VECTOR
__interrupt void USCI_A0_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(USCI_A0_VECTOR))) USCI_A0_ISR (void)
#else
#error Compiler not supported!
#endif
{
  switch(__even_in_range(UCA0IV,4))
  {
  case 0:break;                             // Vector 0 - no interrupt
  case 2:                                   // Vector 2 - RXIFG
    while (!(UCA0IFG&UCTXIFG) || (UCA0STAT&UCBUSY)); // USCI_A0 TX buffer ready?
    write_enable(1);
    UCA0TXBUF = UCA0RXBUF;                  // TX -> RXed character
    while (!(UCA0IFG&UCTXIFG) || (UCA0STAT&UCBUSY));
    write_enable(0);
    break;
  case 4:break;                             // Vector 4 - TXIFG
  default: break;
  }
} (from uart.c)
```

This interrupt routine example reads from the interface and writes the same data back.

## 3.5 Modbus

Modbus is a serial communications protocol originally created for use with Programmable Logic Controllers (PLCs). It is a simple, robust and is now a commonly available means for connecting industrial electronic devices.

Modbus enables communication among approximately 247 devices connected in the same network.

The protocol consists of frames with a delay between each one to separate them. Each frame is composed of an Application Data Unit (ADU) which encloses a Protocol Data Unit (PDU):

- ADU = Address + PDU + Error check
- PDU = Function code + Data

There are 3 frame formats: RTU, ASCII and TCP

### 3.5.1 Modbus frame formats

#### *RTU*

| Name | Length (bits) | Function |
|---|---|---|
| Start | 28 | At least 3.5 character times of silence |
| Address | 8 | Station address |
| Function | 8 | Indicates the function code; e.g., read coils/holding registers |
| Data | n x 8 | Data + length will be filled depending on the message type |
| CRC | 16 | Cyclic Redundancy Check |
| End | 28 | At least 3.5 character times of silesnce between frames |

*Table 10: Modbus RTU frame format*

#### *ASCII*

| Name | Length (bytes) | Function |
|---|---|---|
| Start | 1 | Starts with colon : (ASCII hex value is 0x3A) |
| Address | 2 | Station address |
| Function | 2 | Indicates the function code; e.g., read coils/holding registers |
| Data | n x 2 | Data + length will be filled depending on the message type |
| CRC | 2 | Longitudinal Redundancy Check |
| End | 2 | Carriage return – line feed (CR/LF) pair |

*Table 11: Modbus ASCII frame format*

#### *TCP*

| Name | Length (bytes) | Function |
|---|---|---|
| Transaction identifier | 2 | For synchronization between messages of server & client |
| Protocol identifier | 2 | Zero for Modbus/TCP |
| Length field | 2 | Number of remaining bytes in this frame |
| Unit identifier | 1 | Slave address (255 if not used) |
| Function code | 1 | Function codes as in other variants |
| Data bytes | n | Data as response or commands |

*Table 12: Modbus TCP frame format*

### 3.5.2 Modbus function codes

| Function type | | | Function name | Function code |
|---|---|---|---|---|
| Data Access | Bit access | Physical Discrete Inputs | Read Discrete Inputs | 2 |
| | | Internal Bits or Physical Coils | Read Coils | 1 |
| | | | Write Single Coil | 5 |
| | | | Write Multiple Coils | 15 |
| | 16-bit access | Physical Input Registers | Read Input Registers | 4 |
| | | Internal Registers or Physical Output Register | Read Holding Registers | 3 |
| | | | Write Single Holding Register | 6 |
| | | | Write Multiple Holding Registers | 16 |
| | | | Read/Write Multiple Registers | 23 |
| | | | Mask Write Register | 22 |
| | | | Read FIFO Queue | 24 |
| | File Record Access | | Read File Record | 20 |
| | | | Write File Record | 21 |
| Diagnostics | | | Read Exception Status | 7 |
| | | | Diagnostic | 8 |
| | | | Get Com Event Counter | 11 |
| | | | Get Com Event Log | 12 |
| | | | Report Slave ID | 17 |
| | | | Read Device Identification | 43 |
| Other | | | Encapsulated Interface Transport | 43 |

*Table 13: Modbus function codes*

### 3.5.3 modbus code

#### *Delay*

```
void TimerDelay(unsigned int time)
{
      StartCount(time);
      while (delay);
      TA0CCTL0 = 0;           //disable interupt
}

void StartCount(unsigned int time)
{
      __disable_interrupt();
      delay = 1;
      TA0CTL = TASSEL_2 + ID_2 + MC_1 + TACLR;
                              // SMCLK (11.0592MHz), input divider= /4,
                              // up mode, clear TAR
      TA0CCR0 = time;
      TA0CCTL0 = CCIE;        //enable interupt
      __enable_interrupt();
}(from modbus.c)
```

#### *Write*

```
void send_packet(char bufferlength)
{
      // USCI_A0 TX buffer ready?
      while (!(UCA0IFG&UCTXIFG) || (UCA0STAT&UCBUSY));
            write_enable(1);

      unsigned char i = 0;
      for (i = 0; i < bufferlength; i++)
      {
            UCA0TXBUF = frame[i];
            while (!(UCA0IFG&UCTXIFG));
      }
      while (!(UCA0IFG&UCTXIFG) || (UCA0STAT&UCBUSY));

      //add a frame delay
      TimerDelay(t35);
      write_enable(0);
}  (from modbus.c)
```

The function reads from a buffer and transmits it over the modbus interface.

#### *Read*

Reading is similar to the UART interface.

# 4 Current status of code

All the sensors can be read, and the appropriate values calculates.
The microcontroller can communicate reliably over RS485.

The Modbus protocol needs more work and can use a lot of improvement. But the code shows the prototype is definitely capable of doing all the required tasks.

# PART 4: Conclusion

The selected sensors are a great fit for the project, they are affordable and reliable.

There are of course a lot of potential improvements and future work to be done.

The currently used controller has a lot of unused pins, a smaller cheaper one could be used. Other controllers might also be easier to solder.

The Modbus library needs a lot of work.

A board with a real ozone sensor needs to be made and testing done to see if measuring so far away from potential sparks leads to usable data.

# PART 5: Appendix

# Index of Tables



# Index of images

# Referenced websites

## FIR

http://www.lumasenseinc.com/uploads/Solutions/pdf/Technical_Literature/English/Infrared-Thermometer_Handbook.pdf
http://support.fluke.com/raytek-sales/Download/Asset/IR_THEORY_55514_ENG_REVB_LR.PDF
http://en.wikipedia.org/wiki/Emissivity
http://www.thermoworks.com/emissivity_table.html
http://www.omega.com/temperature/z/pdf/z088-089.pdf
http://www.monarchserver.com/TableofEmissivity.pdf
http://www.digi-key.be
http://www.ti.com
http://www.futurlec.com/PIR_Sensors.shtml
http://www.meas-spec.com/temperature-sensors/thermopile-infrared-sensors-modules/thermopile-infrared-ir-sensors.aspx
http://www.amphenol-sensors.com/en/products/temperature/infrared-ir-sensors
http://www.melexis.com/Infrared-Thermometer-Sensors/Infrared-Thermometer-Sensors/

## Ozone

http://www.hindawi.com/journals/js/2009/608714/
https://www.futurlec.com
http://euro-gasman.com
http://sgx.cdistore.com/PortalMain.aspx
http://www.bw-gasmonitors.com/
http://www.kuntzeinstruments.com
http://uk.alibaba.com
http://www.winsentech.com

## Humidity

http://www.digikey.be
http://sensing.honeywell.com/products/humidity-sensors?Ne=2308&N=3217
http://www.sensorsmag.com/sensors/humidity-moisture/choosing-a-humidity-sensor-a-review-three-technologies-840

https://www.silabs.com/

http://www.meas-spec.com/

## Hardware

http://www.altium.com/
http://www.cadsoftusa.com/
http://www.kicad-pcb.org
https://en.wikipedia.org/wiki/TI_MSP430
http://www.ti.com/tool/msp-exp430f5438
http://www.ti.com/product/MSP430F5438A
http://www.ti.com/product/sn65hvd78/samplebuy
http://www.ti.com/product/sn65hvd75/samplebuy
http://www.urel.feec.vutbr.cz/web_documents/dilna/Vyroba_DPS_2014_03_10.pdf PCB rules
http://www.robotroom.com/PCB-Layout-Tips.html

## Software

http://www.ti.com/tool/ccstudio
https://www.iar.com/iar-embedded-workbench/texas-instruments/msp430/
http://www.rowley.co.uk/msp430/
http://processors.wiki.ti.com/index.php/Printf_support_for_MSP430_CCSTUDIO_compiler
http://www.freemodbus.org/
http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf
http://www.simplymodbus.ca/exceptions.htm
https://github.com/angeloc/simplemodbusng/tree/master/SimpleModbusSlave
https://en.wikipedia.org/wiki/Modbus
http://www.modbusdriver.com/modpoll.html
https://github.com/bashwork/pymodbus

## Datasheets

http://www.melexis.com/Asset/IR-sensor-thermometer-MLX90614-Datasheet-DownloadLink-5152.aspx MLX90614

http://www.ti.com/lit/ug/slau208o/slau208o.pdf MSP4300x5xx and MSP430x6xx Family User's Guide

http://www.ti.com/lit/ds/symlink/sn65hvd78.pdf SN65HVD78

https://www.silabs.com/Support%20Documents/TechnicalDocs/Si7020-A20.pdf Si7020-A20

http://www.ti.com/lit/ds/symlink/msp430f5438a.pdf MSP430f5438A

http://sgx.cdistore.com/datasheets/e2v/1087_Datasheet-MiCS-2614.pdf MiCS-2614